

Application Number: 10/826,584  
Filing Date: 04/16/2004  
Applicant(s): Robert Balzer  
Application Title: By-pass and tampering protection for application wrappers  
Examiner/GAU: Parthasarathy, Pramila / 2136

1/11/04 DAC



**In the United States Patent and Trademark Office**

**Petition to Revive**

Customer Service Window/Mail Stop Petition  
Commissioner for Patents  
Randolph Building  
401 Dulany Street  
Alexandria, VA 22313-1450

Sir/Madam:

On May 19, 2008, I reviewed the Patent Application Information Retrieval webpage for the captioned patent application in order to monitor its progress. Under Status, the webpage indicated "Abandoned – Failure to Respond to an Office Action." A printout of that webpage is attached hereto as Exhibit A. The webpage also indicated the Status Date as March 27, 2008. In response to this information, please be advised that under 37 CFR 1.1137(b), I hereby petition and request that the application be revived on the grounds of UNINTENTIONAL DELAY.

**First Office Action**

A non-final office action was issued to Applicant on 9/12/2007. The office action indicated that Applicant was required to reply within a shortened statutory time period of three months, which time period would expire on 12/12/2007. The office action also indicated that the maximum statutory period is 6 months (which time period would expire on 3/12/2008), and that extensions of time may be available under 37 CFR 1.136(a).

**Request for Extension of Time**

37 CFR 1.136(a)(1) provides that Applicant can extend a maximum of 3 months to 3/12/2008 if petition and fee as specified in 37 CFR 1.17(a) fee are filed. Applicant filed a petition to extended by 2 months to 2/12/2008 by filing a petition and the fee as specified of \$230 on 1/10/2008. A copy of this petition is attached hereto as Exhibit B. This petition was sent via USPS First Class Mail. The petition was mailed to the same address that Applicant mailed the first response to Notice of File Missing Parts of Nonprovisional Application (specifically, Assistant Commissioner for Patents, Washington, DC 20231).

05/23/2008 AWOHDAF1 00000005 10026584

01 FC:2453

770.00 OP

**Reply to First Office Action**

37 CFR 1.136(a)(2) provides that the Applicant reply must be filed prior to the expiration of the period of extension to avoid abandonment of the application. Applicant filed the

Application Number: 10/826,584  
Filing Date: 04/16/2004  
Applicant(s): Robert Balzer  
Application Title: By-pass and tampering protection for application wrappers  
Examiner/GAU: Parthasarathy, Pramila / 2136

reply on 2/8/2008, prior to the period of extension expiration of 2/12/2008. A copy of the reply is attached hereto as Exhibit C. The reply was sent via Federal Express to United States Patent and Trademark Office, Customer Service Window / Mail Stop Amendment, Randolph Building, 401 Dulany Street, Alexandria, VA 22314. A copy of the Federal Express bill evidencing this shipment is attached as Exhibit D.

#### **Return of Request for Extension of Time**

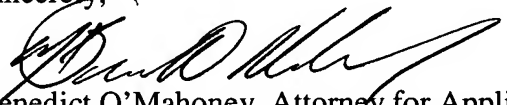
The USPS returned the Request for Extension of Time to Applicant on March 4, 2008, indicating that the address was not correct. Applicant immediately placed the returned package unopened, with all stamps and markings placed by the USPS intact, into a new envelope and sent the package via Federal Express to Customer Service Window Mail Stop A, U.S. Patent and Trademark Office, Randolph Building, 401 Dulany Street, Alexandria, VA 22314. A copy of the Federal Express tracking receipt is attached hereto as Exhibit E. The returned package was received by the USPTO on March 5, 2008. The check representing the fee required pursuant to 37 CFR 1.17(a) for the extension was cashed on March 12, 2008. A copy of the endorsed check is attached hereto as Exhibit F.

#### **Petition to Revive**

As evidenced in the foregoing, the Request for Extension of Time was ultimately timely made, and the reply was made within the time period requested by the Request for Extension of Time. The entire delay in filing the required reply from the due date for the reply was unintentional. For all of the foregoing, Applicant hereby petitions that the captioned application be revived. As required by 37 CFR 1.137(b)(2), enclosed herewith is a check in the amount of \$770.00 representing the petition fee as set forth in 37 CFR 1.17(m).


Thank you for your assistance with the application.

Sincerely,

  
Benedict O'Mahoney, Attorney for Applicant

I hereby certify that this correspondence is being deposited with the Federal Express as overnight delivery mail on the date indicated below in an envelope addressed to:

United States Patent and Trademark Office  
Customer Service Window, Mail Stop Petition  
Randolph Building, 401 Dulany Street  
Alexandria, VA 22314

  
Benedict O'Mahoney

5/21/2008  
Date

Application Number: 10/826,584  
Filing Date: 04/16/2004  
Applicant(s): Robert Balzer  
Application Title: By-pass and tampering protection for application wrappers  
Examiner/GAU: Parthasarathy, Pramila / 2136

**EXHIBIT A**



## United States Patent and Trademark Office

[Home](#) | [Site Index](#) | [Search](#) | [FAQ](#) | [Glossary](#) | [Guides](#) | [Contacts](#) | [eBusiness](#) | [eBiz Alerts](#) | [News](#) | [Help](#)
[Portal Home](#) | [Patents](#) | [Trademarks](#) | [Other](#)
Patent eBusiness ☐ Patent Application Information Retrieval ☐

- ☐ [Electronic Filing](#)
- ☐ [Patent Application Information \(PAIR\)](#)
- ☐ [Patent Ownership](#)
- ☐ [Fees](#)
- ☐ [Supplemental Resources & Support](#)

## Patent Information

## Patent Guidance and General Info

- ☐ [Codes, Rules & Manuals](#)
- ☐ [Employee & Office Directories](#)
- ☐ [Resources & Public Notices](#)

## Patent Searches

## Patent Official Gazette

- ☐ [Search Patents & Applications](#)
- ☐ [Search Biological Sequences](#)
- ☐ [Copies, Products & Services](#)

## Other

[Copyrights](#)  
[Trademarks](#)  
[Policy & Law](#)  
[Reports](#)

☐ [Order Certified Application As Filed](#) [Order Certified File Wrapper](#) [View Order List](#)

10/826,584

By-pass and tampering protection for application wrappers



Select New Case	Application Data	Transaction History	Image File Wrapper	Continuity Data	Published Documents	Address & Attorney/Agent
-----------------	------------------	---------------------	--------------------	-----------------	---------------------	--------------------------

## Bibliographic Data

Application Number:	10/826,584	Customer Number:	-
Filing or 371 (c) Date:	04-16-2004	Status:	Abandoned -- Failure to Respond to an Office Action
Application Type:	Utility	Status Date:	03-27-2008
Examiner Name:	PARTHASARATHY, PRAMILA	Location:	ELECTRONIC
Group Art Unit:	2136	Location Date:	-
Confirmation Number:	2981	Earliest Publication No:	US 2005-0108516 A1
Attorney Docket Number:	2004-001	Earliest Publication Date:	05-19-2005
Class / Subclass:	713/150	Patent Number:	-
First Named Inventor:	Robert Balzer , Sante Fe, NM	Issue Date of Patent:	-

Title of Invention: By-pass and tampering protection for application wrappers

## If you need help:

- Call the Patent Electronic Business Center at (866) 217-9197 (toll free) or e-mail [EBC@uspto.gov](mailto:EBC@uspto.gov) for specific questions about Patent Application Information Retrieval (PAIR).
- Send general questions about USPTO programs to the [USPTO Contact Center \(UCC\)](#).
- If you experience technical difficulties or problems with this application, please report them via e-mail to [Electronic Business Support](#) or call 1 800-786-9199.

You can suggest USPTO webpages or material you would like featured on this section by E-mail to the [webmaster@uspto.gov](mailto:webmaster@uspto.gov). While we cannot promise to accommodate all requests, your suggestions will be considered and may lead to other improvements on the website.

[Home](#) | [Site Index](#) | [Search](#) | [eBusiness](#) | [Help](#) | [Privacy Policy](#)

Application Number: 10/826,584  
Filing Date: 04/16/2004  
Applicant(s): Robert Balzer  
Application Title: By-pass and tampering protection for application wrappers  
Examiner/GAU: Parthasarathy, Pramila / 2136

**EXHIBIT B**



**In the United States Patent and Trademark Office**

Serial Number: 60/463,770  
Application Filed: 04/17/2003  
Applicant(s): Robert Balzer  
Application Title: By-pass and tampering protection for application wrappers  
Examiner/GAU:

**Request for Extension of Time**

Assistant Commissioner for Patents  
Washington, D.C. 20231

Sir:


It is respectfully requested that an extension of time for the period indicated below be granted in accordance with the provisions of 37 C.F.R. Section 1.136 to take action required in the application identified in the caption, as reflected by the papers submitted herewith:

Selected:			Small Entity Fee
	Extension for response within first month	120.00	60.00
X	Extension for response within second month	460.00	230.00
	Extension for response within third month	1,050.00	525.00
	Extension for response within fourth month	1,640.00	820.00
	Extension for response within fifth month	2,230.00	1,115.00

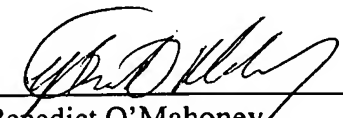
A check in the amount of \$230.00 is attached. This amount is believed to be correct.

Thank you for your assistance with the application.

Sincerely,

  
Benedict O'Mahoney, Attorney for Applicant

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addresses to: Assistant Commissioner for Patents, Washington, D.C. 20231, on the date indicated below.

 \_\_\_\_\_  
Benedict O'Mahoney

\_\_\_\_\_ 4/10/2008  
Date

UR REFERENCE NUMBER	YOUR INVOICE NUMBER	INVOICE DATE	INVOICE AMOUNT	AMOUNT PAID	DISCOUNT	NET AMOUNT
32011	010208	01/02/08	230.00	230.00	0.00	230.00
						----- \$230.00



TEKNOLEDGE

TEKNOLEDGE

CORPORATE HEADQUARTERS  
1800 EMBARCADERO ROAD  
PALO ALTO, CALIFORNIA 94303

Bridge Bank  
2120 El Camino Real  
Santa Clara, CA 95050  
www.bridgebank.com

90-4328/1211

54620

DATE

CONTROL NO.

AMOUNT

01/10/08

54620

\*\*\*\*230.00\*\*

\*\* Two Hundred Thirty Dollars And 0 Cents

PAY  
TO THE  
ORDER OF

US Patent And Trademark Office  
2011 South Clark Place  
Crystal Plaza 2, Lobby, Rm 1B03  
Arlington VA 22202

*Michael D. Kayser*  
*[Signature]*

TEKNOLEDGE

Application Number: 10/826,584  
Filing Date: 04/16/2004  
Applicant(s): Robert Balzer  
Application Title: By-pass and tampering protection for application wrappers  
Examiner/GAU: Parthasarathy, Pramila / 2136

**EXHIBIT C**





**In the United States Patent and Trademark Office**

Serial Number: 60/463,770  
Application Filed: 04/17/2003  
Applicant(s): Robert Balzer  
Application Title: By-pass and tampering protection for application wrappers  
Examiner/GAU:

**AMENDMENT A**

United States Patent and Trademark Office  
Customer Service Window, Mail Stop Amendment  
Randolph Building  
401 Dulany Street  
Alexandria, VA 22314

Sir:

In response to the office action mailed September 12, 2007, please amend the above application as follows:

**Specification:**

The following amendments are made in the context of clarifying the trademark issues pertaining to Microsoft and Intel.

Page 6, the description for Figure 3 is replaced with the following description:

Figure 3 illustrates the security of the Windows System Service Handling Protocol.

Page 9, second paragraph, replace with the following new paragraph:

Regarding the native operating system service invocation protocol, Figure 2 depicts the native Microsoft® Windows operating system software architecture for secure employment of System Services by application code.

Page 9, last paragraph, replace with the following new paragraph:

In practice, applications do not deal with this machine language interface. Instead, they invoke functions exported from Microsoft®-supplied libraries, which directly, or indirectly through other Microsoft®-supplied libraries, invoke code that actually writes the registers and runs the int2E instruction.

Page 10, first paragraph, replace with the following new paragraph:

The int instruction treats its operand as an index into a vector called the Interrupt Descriptor Table [105]. The members of this vector are Gate Descriptors. The address of this table is kept in a register that cannot be written by user-mode code. As part of booting, the Windows operating system sets this register to point to a block of memory that cannot be modified by user-mode code, and fills the entry indexed by 2E with values that comprise an Interrupt Gate Descriptor. Interrupt gates are one of several mechanisms provided by the Intel® microprocessor for switching privilege levels. Although the processor provides four levels, the Windows operating system only makes use of two of them – Level 3 (aka User Mode) and Level 0 (aka Kernel Mode). The data written by the bootstrapping code into entry 2E specifies a transfer to Kernel Mode, and the address of a function in the Windows operating system kernel (the system service dispatcher (aka KiSystemService) [107]) to which control should be transferred. The int instruction copies the values in several hardware registers to the kernel stack [108] for the calling thread (so they can be restored on return), as well. Among these are the stack pointer register (ESP) containing the address of the top of the thread's user-mode stack, and the instruction pointer (EIP)

register, containing the address of the instruction following the int2E instruction [109].

Page 10, last paragraph (extends to page 11), replace with the following new paragraph:

The CPU is now running at privilege level 0, executing the system service dispatcher. This code treats the service number (still present in the EAX register) as an index into a System Service Dispatch Table [110], a vector in kernel memory allocated and initialized during the bootstrapping of the Windows operating system. The low-order 12 bits of the service number act as an index into a dispatch table; the next two higher-order bits select one of four possible tables. In practice, one table is used for a set of services implemented in the binary win32k.sys, which contains user interface related services. A second table is used for the services implemented in ntoskrnl.exe. The indexed entry provides KiSystemService with the address of the kernel function that implements the requested service and how many parameter bytes are expected by that function. If the index is valid, KiSystemService copies the parameters from user mode memory (the base address to copy from is still in the EDX register) onto the kernel stack [112], and then calls the designated kernel function.

Page 13, first paragraph, replace with the following new paragraph:

The contents of the Interrupt Descriptor Table [105] and the System Service Dispatch Table [110] reside in kernel-mode memory and so are safe from tampering. The register holding the address of the Interrupt Descriptor Table can only be written by kernel mode code. CPU registers, which will hold data and

instruction addresses during the execution of the Service Handler, are properly saved and restored if the operating system decides to perform a thread switch while the thread is in kernel mode.

Page 13, third paragraph, replace with the following new paragraph:

In the augmented implementation and protocol, the Bypass Protection and Tampering Protection are enabled by the addition of two components to the operating system kernel and use of a protocol by application code to establish trusted execution intervals in specific execution threads. Figure 1 depicts these components and the protocol.

Page 13, last paragraph (extends to page 14), replace with the following new paragraph:

The Interrupt Gate Descriptor in entry 2E of the Interrupt Descriptor Table [105] is modified so that its target address is a new piece of kernel-resident code, the System Service Dispatch Tap [106]. Each service request is therefore intercepted by the System Service Dispatch Tap, which checks the Thread Trust Datastore [116] to see if the calling thread is currently trusted to invoke the requested service (the service is identified by the value in the EAX register). If the thread is trusted, the System Service Dispatch Tap transfers control, via a jump (jmp) instruction, to the native System Service Dispatcher [107] (whose address was replaced in Interrupt Gate Descriptor 2E). The System Service Dispatch Tap ensures that the stack pointer, the stack, and all registers contain the same values they held at the time it intercepted the request. This ensures that the System Service Dispatcher will see the same context it would have seen had the

System Service Dispatch Tap not been inserted. If the calling thread is not currently trusted to invoke the requested service, the System Service Dispatch Tap will kill the calling thread or process (if so configured) or return an error result to the caller (e.g., by passing control to the System Service Dispatcher as in an approved request, but with an invalid service index in the EAX register).

Page 14, last paragraphs, replace with the following new paragraph:

A kernel-mode driver not present in the native operating system, the Bypass Driver [115], accepts DeviceIoControl calls from user-mode code. The per-process logic of the Bypass Driver, which is identical for each process, implements the simple state diagram depicted in Figure 4. The per-thread logic of the Bypass driver implements a second simple state machine, depicted in Figure 5. The latter state machine determines the dynamic trust of a thread with respect to system services by modifying the contents of the Thread Trust Database [116]. Figure 6 depicts the detailed logic within the Bypass Driver that implements these state machines.

Page 16, third paragraphs, replace with the following new paragraph:

This protocol requires a properly programmed thread to contain pairings of TrustMe and SuspectMe control codes. These pairings may nest. A programmer who writes the code segment

```
DeviceIoControl(hBypassDriver,TrustMe,...);  
  
TrustedCode();  
  
DeviceIoControl(hBypassDriver,SuspectMe,...);
```

is authorizing a particular invocation of the subroutine TrustedCode to execute with no restrictions on its use of system services (beyond those imposed by the native operating system itself). Although programs typically pair TrustMe/SuspectMe control codes at the same lexical program scope, the logical pairing is purely dynamic and bears no necessary connection to program lexical scope. The execution of code in a thread between a TrustMe/SuspectMe pair is referred to as a trusted interval.

Page 17, last paragraph (extends to page 18), replace with the following new paragraph:

The binary code portion of a program's address space generally comprises code memory segments from multiple executable files, including one or more libraries supplied by the operating system vendor. The "data" portion of the address space includes both "globally" allocated storage segments from these files, "heap" allocated storage allocated "on-demand" by the operating system as the program runs, "stack" storage allocate by the operating system as threads are created, and memory-mapped files. In addition to this virtual memory, programs use the microprocessor's hardware registers as temporary storage and rely on these being "tamper-proof" as well. The Windows operating system allows individual pages of virtual memory to be placed in a write-protected state. While in this state, any attempt by a program to alter the contents of that memory page will fail, and an exception will be raised. The Windows operating system provides a user-mode library utility, VirtualProtect, to change the read/write/execute

protection status of a contiguous range of virtual memory pages. The following steps are required to protect trusted code and its data:

Page 19, third paragraph, replace with the following new paragraph:

Whenever it is necessary to allocate (additional) memory identified in step 3, allocate it on the heap created in step 4 for that partition. The Windows operating system provides the HeapAlloc API for this purpose, but some programming languages (notably C++) provide linguistic means (operators, templates) that can be used to encapsulate an association between types and heaps, which commonly matches the partitioning identified in step 3.

Page 20, last paragraph, replace with the following new paragraph:

In regards to securing the Bypass Driver and Tampering Protector, although the kernel augmentation and protocol usage described above allow a “cooperative” program to dynamically control system service access in its threads, it does not prevent malicious code in a program from accessing those services in unintended ways. Likewise, the Tampering Protection, as described above, allows cooperative code to limit its own updates to sensitive memory, but does not in itself prevent malicious code from gaining update access to that memory. The remaining portion of the Bypass Protector constitutes a means of dealing with the same security concerns addressed by the operating system’s system service call architecture.

Page 24, last paragraph, replace with the following new paragraph:

For most practical purposes, it is necessary for logic in a trusted interval to rely on the contents of machine registers written by code executing in untrusted

intervals. For example, if trusted interval code can be used from multiple threads of a process, it is generally necessary to provide parameters to it on the thread's user-mode stack, which means the code relies on the content of the stack pointer register. Even for single-threaded applications, it is generally more convenient to supply parameters on the stack than in global variables. Malicious code written by a programmer who is aware of how trusted code relies on specific registers can load a register with a value that will cause the trusted code to throw an exception, prior to jumping directly to the inline TrustMe code. If done properly, the thread would enter trusted mode and then raise an exception. If the malicious code established its own SEH exception handler in the calling thread prior to this, it could gain control while the thread is still in a trusted state, and therefore employ guarded services without passing through the intended validity checking. To prevent this, the user-mode code must disable existing SEH handlers in the calling thread after the TrustMe request returns (i.e., following the int 2E instruction in the inline-coded version), but before executing any code that could raise an exception due to improper register contents. For an Intel® CPU, the assembly language instruction:

Page 25, last paragraph (extends to page 26), replace with the following new paragraph:

Use of the Bypass Protector to block access to system services from dynamically loaded and executed untrusted code in the above scenario is quite limiting, because it blocks all access to those services, independent of context or parameters. This motivates the second usage scenario, in which the trusted code



acts as a filter on requests for the guarded services originating in untrusted code. Filtering is accomplished by mediating (rerouting) the indirect requests for those services originating in the untrusted code. The untrusted code makes an indirect request by invoking an API exported from a user-mode library. (In practice, these indirect requests are the only way that non-malicious applications actually invoke system services on the Windows operating system.) The low-level means of dynamically patching binary code in virtual memory to accomplish this mediation are in the public domain – the article “Galen C. Hunt, Doug Brubaker, Detours: Binary Interception of Win32 Functions Proceedings of the 3rd USENIX Windows NT Symposium, July 1999, pp. 135-43.” contains an excellent explanation. In this scenario, the API calls from untrusted code are rerouted to code that begins a trusted interval, inspects the context and API call parameters to determine whether to approve or reject the call, and (conditional on approval) routes invokes the original API by a means that won’t be rerouted. When the latter call returns, the trusted code closes the trusted interval.

Page 27, first paragraph, replace with the following new paragraph:

With regards to establishing a Bypass and/or Tamper Protected Process, the first step of establishing a protected process is to install the Bypass Driver, which initializes the kernel-resident portion of the architecture. Typically the driver is included with other drivers auto-installed when the Windows operating system boots, but it could be installed anytime prior to beginning the following steps.

Page 27, second paragraph, replace with the following new paragraph:

The trusted module of the protected process may be loaded into the process before or after the untrusted portion. The two scenarios described above present an example of each. The trusted module should do the following, in the order listed:

Obtain from the operating system (via the CreateFile API), and retain in virtual memory, a handle to the Bypass Driver. This is needed to send DeviceIoControl requests to the driver.

Page 27, third paragraph, replace with the following new paragraph:

Send the GuardServices code to the driver, passing an array of system service indices to be guarded. The index of a particular system service may depend on which version of the operating system is being used, so for a trusted code binary to be version-independent it must use GetVersionEx to determine the platform version.

The following amendments are made in the context of clarifying the references to the Thread Trust Datastore.

Page 7, the description for Drawings – Reference Numeral 116 is replaced with the following description:

116 – Thread Trust Datastore

Page 12, last paragraph, replace with the following new paragraph:

All the kernel-mode code, including that comprising ValidForThisThread and PerformService, is protected from being overwritten by instructions executed in application code. This is because the protected code resides kernel-mode memory segments, and the CPU prevents user mode code from altering that

memory. Data on which the service relies [116], such as access control lists, thread account ids, and the thread's kernel stack reside in a thread trust datastore which is located in kernel-mode memory and so are safe from tampering.

The following amendments are made in the context of clarifying the references to the System Service Dispatch Table, System Service Dispatcher and System Service Dispatch Tap by replacing acronyms with full names or removing superfluous acronyms.

Page 7, the description for Drawings – Reference Numeral 106 is replaced with the following description:

106 – System Service Dispatch Tap

Page 13, first paragraph, replace with the following new paragraph:

The contents of the Interrupt Descriptor Table [105] and the System Service Dispatch Table [110] reside in kernel-mode memory and so are safe from tampering. The register holding the address of the Interrupt Descriptor Table can only be written by kernel mode code.

Page 15, first paragraph, replace with the following new paragraph:

A newly created process is (initially) unrestricted [123]. All threads of an unrestricted process are treated as trusted for all services. This is the interpretation of having no entry for a thread's process in the Thread Trust Datastore.

Page 15, second paragraph, replace with the following new paragraph:

The first time any thread of a process sends the GuardServices control code [124,146], a set of guarded services is established for all threads (current and future) of the calling process. The GuardServices control code is accompanied by an array of service numbers. An entry is added to the Thread Trust Datastore for

the calling process, containing a record of the service numbers to be guarded. The process is now in the restricted state [125,134]. All current threads of the process are treated as untrusted for the guarded services, but trusted for all other services. This is the interpretation of the Thread Trust Datastore for a thread that has no entry in the Thread Trust Datastore when its process is restricted. A consequence of this treatment is that any newly created threads of a restricted process are initially untrusted for the guarded services.

Page 15, third paragraph, replace with the following new paragraph:

When a thread of a restricted process sends a TrustMe [130/132,137] control code, that thread's trust count in the Thread Trust Datastore is incremented by one [138]. If the thread has no entry in the Thread Trust Datastore [140], one is created with an initial trust count of 0 [145], then incremented to 1 [138].

Page 15, fourth paragraph, replace with the following new paragraph:

When a thread with an entry in the Thread Trust Datastore having a trust count greater than zero sends a SuspectMe [130/133, 141] control code, its entry's trust count is decremented by one [142].

Page 16, first paragraph, replace with the following new paragraph:

All other DeviceIoControl control codes to the Bypass Driver are invalid [144]. Depending on configuration, the calling thread or process will be terminated, or the caller will simply receive an error result and have no impact on the Thread Trust Datastore.

Page 16, second paragraph, replace with the following new paragraph:

The Bypass Driver registers with the operating system kernel (PsSetCreateProcessNotifyRoutine) to be notified of process termination. When a process is terminated, all records involving that process and its threads are removed from the Thread Trust Datastore. The Bypass Driver registers with the operating system kernel (PsSetCreateThreadNotifyRoutine) to be notified of thread termination. When a thread is terminated, all records involving that thread are removed from the Thread Trust Datastore.

Page 21, first paragraph, replace with the following new paragraph:

Code such as TrustedCode() in the fragment above can be written as:

if ValidForThisThread(Parameters) then PerformService(Parameters)

where PerformService invokes one of the process's guarded services with assurance that malicious code cannot execute PerformService directly by transferring control to it. If a direct transfer were made, the System Service Dispatch Tap would reject the use of guarded services by PerformService, because it would occur while the calling thread was not trusted. The user-mode code, containing ValidForThisThread and PerformService can be protected from being overwritten by malicious user-mode code by placing tampering protection on the memory segments containing the code. User-mode data on which trusted code relies is held in memory segments with tampering protection.

Page 21, second paragraph, replace with the following new paragraph:

The contents of the Thread Trust Datastore is safe from corruption by user-mode code because it resides in kernel memory. Likewise, the code

comprising the Bypass Driver and the System Service Dispatch Tap are safe from corruption because they reside in kernel memory.

Page 21, last paragraph (extends to page 22), replace with the following new paragraph:

The contents of the Thread Trust Datastore is safe from corruption by user-mode code because it resides in kernel memory. Likewise, the code comprising the Bypass Driver and the System Service Dispatch Tap are safe from corruption because they reside in kernel memory. Nothing prevents a user-mode program from writing and executing its own sequence of instructions:

```
DeviceIoControl(hBypassDriver,TrustMe,...);
```

```
TrustedCode();
```

```
DeviceIoControl(hBypassDriver,SuspectMe,...);
```

in user memory. To thwart this, we extend the Bypass Driver protocol and its System Service Dispatch Tap functionality as follows.

Page 22, second paragraph; replace with the following new paragraph:

An additional control code, TrustInitializers, is accepted by the Bypass Driver [126]. The driver will accept only the first request sent by a process using this code, and it must follow the GuardServices request from the same process. The data accompanying the TrustInitializers code is an array of 32-bit user mode memory addresses. These are the addresses of the instructions following the "int 2Eh" instructions - one for each open-coded TrustMe call in the program. The array of addresses is associated with the process in the Thread Trust Datastore.

Upon receipt of this code, the driver places the process in the ready state in the Thread Trust Datastore [127].

Page 22, last paragraph (extends to page 23), replace with the following new paragraph:

A minor augmentation of the logic in Figure 6 implements the extra transition in the state diagram. The TrustInitializers code transitions a process's state from "Restricted" to "Ready". TrustMe codes are now only accepted in the protocol for threads of processes whose state is Ready, not those whose state is restricted. When the Bypass driver receives a TrustMe control code from a process, it verifies that the return address (RA) is one of those associated with the calling process in the Thread Trust Datastore. If it is not, the request is considered a "spoofing" attempt. Depending on the configuration, the requesting thread or process will be terminated, or the requester will simply receive an error result and the TrustMe request will not be granted.

**Claims:**

Please replace the entire set of pending claims (claims 1-8) with the following clean set, pursuant to 37 C.F.R. §1.121(c)(3):

1. A computer system including an operating system and software applications, the system comprising:

a central processing unit;

means for storing and retrieving programs and data connected with said central processing unit;

an operating system stored in said means for storing and retrieving programs and data;

a plurality of software applications stored in said means for storing and retrieving programs and data;

a plurality of application threads, wherein each of said threads is associated with a single one of said software applications;

a plurality of bypass protocols that interface with said software applications, wherein each of said bypass protocols is associated with a single one of said software applications;

a bypass driver that interfaces with said bypass protocols, wherein the specific state of trust of each of said application threads of said software applications associated with said bypass protocols is obtained by said bypass driver from said bypass protocols;

a thread trust datastore that interfaces with said bypass driver, wherein the state of trust of said application threads of said software applications is communicated from said bypass driver to said thread trust datastore and stored in memory; and

a system service dispatch tap that interfaces with said operating system, wherein invocations of services from said operating system by said software applications are intercepted by said system service dispatch tap, the state of trust of said application threads of said software application is obtained from said thread trust datastore, and said invocation of service is routed in said operating system based upon said state of trust.



2. The computer system including an operating system and software applications of claim 1, wherein said operating system is a Microsoft® Windows® based operating system.

3. The computer system including an operating system and software applications of claim 1, wherein said central processing unit is an Intel®-based microprocessor.

4. The computer system including an operating system and software applications of claim 1, wherein said invocation of service is terminated if the value of said state of trust of said application thread of said software application is negative.

5. A computer system including an operating system and software applications, the system comprising:

a central processing unit;

means for storing and retrieving programs and data connected with said central processing unit;

an operating system stored in said means for storing and retrieving programs and data;

a plurality of privilege levels associated with said central processing unit;

a plurality of software applications stored in said means for storing and retrieving programs and data, wherein each of said applications is associated with a single one of said privilege levels;

a plurality of application threads, wherein each of said application threads is associated with a single one of said software applications;

a plurality of driver modules stored in said means for storing and retrieving programs and data, wherein each of said driver modules are associated with a single one of said privilege levels;

a plurality of return addresses, wherein each of said return addresses are associated with a single one of said software applications;

a plurality of driver requests, wherein each of said driver requests is associated with a single one of said software applications and a single one of said return addresses, and said software application associated with each of said driver requests is associated with a lower privilege level than the privilege level associated with the driver to which said request is directed;

a plurality of bypass protocols that interface with said software applications, wherein each of said bypass protocols is associated with a single one of said software applications and a single one of said driver modules;

a thread trust datastore that interfaces with said driver modules, wherein the return addresses of said software applications are obtained by said driver modules and stored in said thread trust datastore, and one of said return addresses associated with one of said software applications may subsequently be retrieved by said driver modules, compared with one of said return addresses associated with one of said driver requests from one of said software applications, and said driver request is routed differentially based on whether said return address associated with said driver request is associated in said driver module's thread trust datastore with the requesting thread's application.

6. The computer system including an operating system and software applications of claim 5, wherein said driver request is denied if said return address of said software application does not match one of said return addresses associated with said calling thread's software application in said thread trust datastore.

7. The computer system including an operating system and software applications of claim 5, wherein said operating system is a Microsoft® Windows® based operating system.

8. The computer system including an operating system and software applications of claim 5, wherein said central processing unit is an Intel®-based microprocessor.

**Version of Claims with Markings to Show Changes Made:**

1. A computer system including an operating system and software applications, the system comprising:
  - a central processing unit;
  - means for storing and retrieving programs and data connected with said central processing unit;
  - an operating system stored in said means for storing and retrieving programs and data;
  - a plurality of software applications stored in said means for storing and retrieving programs and data;
  - a plurality of application threads, wherein each of said threads is associated with a single one of said software applications;

a plurality of bypass protocols that interface with said software applications, wherein each of said bypass protocols is associated with a single one of said software applications;

a bypass driver that interfaces with said bypass protocols, wherein the specific state of trust of each of said application threads of said software applications associated with said bypass protocols is obtained by said bypass driver from said bypass protocols;

a thread trust datastore that interfaces with said bypass driver, wherein the state of trust of said application threads of said software applications is communicated from said bypass driver to said thread trust datastore and stored in memory; and a system service dispatch tap that interfaces with said operating system, wherein invocations of services from said operating system by said software applications are intercepted by said system service dispatch tap, the state of trust of said application threads of said software application is obtained from said thread trust datastore, and said invocation of service is routed in said operating system based upon said state of trust.

2. The computer system including an operating system and software applications of claim 1, wherein said operating system is a Microsoft® Windows® based operating system.
3. The computer system including an operating system and software applications of claim 1, wherein said central processing unit is an Intel®-based microprocessor.

4. The computer system including an operating system and software applications of claim 1, wherein said invocation of service is terminated if the value of said state of trust of said application thread of said software application is negative.

5. A computer system including an operating system and software applications, the system comprising:

a central processing unit;

means for storing and retrieving programs and data connected with said central processing unit;

an operating system stored in said means for storing and retrieving programs and data;

a plurality of privilege levels associated with said central processing unit;

a plurality of software applications stored in said means for storing and retrieving programs and data, wherein each of said applications is associated with a single one of said privilege levels;

a plurality of application threads, wherein each of said application threads is associated with a single one of said software applications;

a plurality of driver modules stored in said means for storing and retrieving programs and data, wherein each of said driver modules are associated with a single one of said privilege levels;

a plurality of return addresses, wherein each of said return addresses are associated with a single one of said software applications;

a plurality of driver requests, wherein each of said driver requests is associated with a single one of said software applications and a single one of said return

addresses, and said software application associated with each of said driver requests is associated with a lower privilege level than the privilege level associated with the driver to which said request is directed;

a plurality of bypass protocols that interface with said software applications, wherein each of said bypass protocols is associated with a single one of said software applications and a single one of said driver modules;

a thread trust datastore that interfaces with said driver modules, wherein the return addresses of said software applications are obtained by said driver modules and stored in said thread trust datastore, and one of said return addresses associated with one of said software applications may subsequently be retrieved by said driver modules, compared with one of said return addresses associated with one of said driver requests from one of said software applications, and said driver request is routed differentially based on whether said return address associated with said driver request is associated in said driver module's thread trust datastore with the requesting thread's application.

6. The computer system including an operating system and software applications of claim 5, wherein said driver request is denied if said return address of said software application does not match one of said return addresses associated with said calling thread's software application in said thread trust datastore.

7. The computer system including an operating system and software applications of claim 5, wherein said operating system is a Microsoft® Windows based operating system.

8. The computer system including an operating system and software applications of claim 5, wherein said central processing unit is an Intel®-based microprocessor.

### **REMARKS**

The amendment to the description for Figure 3 on page 6, as well as the amendments to the second paragraph of page 9, the last paragraph of page 9, the first paragraph of page 10, the last paragraph of page 10, the first paragraph of page 13, the third paragraph of page 13, the last paragraph of page 13, the last paragraph of page 14, the third paragraph of page 16, the last paragraph of page 17, the third paragraph of page 19, the last paragraph of page 20, the last paragraph of page 24, the last paragraph of page 25, and the first paragraph of page 27, the second paragraph of page 27, and the third paragraph of page 27, address all trademark issues pertaining to Microsoft and Intel.

Claim 3 has been amended to correct the informality by replacing the recitation "microprosser" with "microprocessor".

Claims 1-8 have been amended to define the invention more particularly and distinctly so as to overcome the technical rejections and define the invention patentability over the prior art. Applicants respectfully request further examination and allowance of the application, as amended.

### **The Objection To The Specification And The Claims Rejection Under § 112**

The specification was objected to under § 112 since it was said to fail to describe "the state of trust of software applications", a term that was recited in claims 1 and 4. As indicated above, claims 1 and 4 are amended to more specifically recite that the state of trust pertains to the application threads of the software applications. The determination

of the trust state of the application thread is described on page 14 wherein it is indicated that if the calling thread is not currently trusted to invoke the requested service, the System Service Dispatch Tap will kill the calling thread. Page 15 describes threads as initially untrusted for guarded services, but trusted for all others, but that a control code can change the state of a thread by incrementing or decrementing a trust count in the thread trust datastore.

Applicants request reconsideration and withdrawal of this objection since, in light of the above described amendment, the specification describes a state of trust of said application threads of said software applications in the context of claims 1-4.

Also mentioned is the recitation of a "thread trust datastore" in the context of claims 1-8. Regarding the thread trust datastore, the description for Drawings – Reference Numeral 116 on page 7, last paragraph of page 12, and the last paragraph of page 13 (extending to page 14) were amended to reconcile slight variations to the description of the thread trust datastore. The description on page 7 referred to drawing 116 as a "Trust Store Datastore", and the last paragraph of page 13 (extending to page 14) referred to drawing 116 as a "Thread Trust Database", whereas the actual drawing of 116 in Figure 1 is labeled a "Thread Trust Datastore". Consequently, the description on page 7 and the last paragraph of page 13 (extending to page 14) have been reconciled to the label of the actual drawing. Furthermore, the last paragraph of page 12 is amended to further clarify where in the kernel-mode memory that the thread's kernel stack resides, namely in a thread trust datastore.



Applicants request reconsideration and withdrawal of this objection since, in light of the above described reconciliation, the specification describes a “thread trust datastore” in the context of claims 1-8.

Accordingly, Applicants submit that the specification does comply with § 112 and therefore request withdrawal of this objection.

### **The Rejection Of Claims 1-8 On Proudler Is Overcome**

The examiner rejects claims 1-8 under 35 U.S.C. §102(e) as being unpatentable over Proudler (U.S. Patent No. 20030226031 A1).

Applicants’ invention, as set forth in independent claims 1 and 4, describes a system operating in a standard environment in which a standard operating system is running on a standard computer. In this environment, software code is installed on the standard computer which mediates communications between software applications and the operating system. It provides for trusted communications between the software application and operating system by keeping track of each software application thread, and for each invocation of a service requested by each thread, testing to determine if that thread is trusted before invoking a service.

Applicant’s invention is far different from the system described in Proudler. Proudler describes an apparatus and method for creating a trusted environment [0001] by modifying a computer by installing a virtual trusted device, consisting of a layer of hardware and software [0037], between the CPU and the operating system. In this environment, the virtual trusted device performs a validity check on the computer and compares the data with data from a trusted third party [0038]. If there is a match, the environment is considered trusted, and the operating system is allowed to run in a

modified privilege level [0042]. The system described in Proudler only works if the virtual trusted device takes control before the computer boots up: "In the event the trusted device 24 is not the first accessed, there is of course a chance that the trusted device 24 will not be accessed at all. This would be the case, for example, if the main processor 21 were manipulated to run the BIOS program first. Under these circumstances, the platform would operate, but would be unable to verify its integrity on demand, since the integrity metric would not be available. Further, if the trusted device 24 were accessed after the BIOS program had been accessed, the Boolean value would clearly indicate lack of integrity of the platform." [0074]

Nothing in Proudler teaches or suggests the invention as set forth in Applicants' claims. Claims 1-8 have novel (and unobvious) features. Specifically, claim 1 describes "a plurality of bypass protocols that interface with said software applications, wherein each of said bypass protocols is associated with a single one of said software applications;" The only interface to software applications referenced in Proudler pertain to the description that the "end user applications 36 run at the least privileged level, PL3, as unprivileged tasks under the control of an operating system image 35 in a secure platform 34 protection domain." [0059] Proudler goes on the state that "[I]n order for the computer platform 10 and operating environment(s) to be trusted, a chain of trust from the system hardware, through the boot process, to final running code is established." [0060] In order to accomplish this, Proudler teaches that "[T]he chain of trust extends back to the trusted device 24. As described below, after system reset the processor 21 is initially controlled by the trusted device 24, which then after performing a secure boot process hands control over to the BIOS firmware 28. During the secure boot process, the

trusted device 24 acquires an integrity metric of the computer platform 10, as described below.” Here, there is no indication that individual software applications can be trusted to operate in an environment that has not been created on a secure platform created by a device that has intercepted the computer boot process, nor is there any teaching that individual software application can be associated with bypass protocols.

Furthermore, claim 1 describes “a bypass driver that interfaces with said bypass protocols, wherein the specified state of trust of each of said application threads of said software applications associated with said bypass protocols is obtained by said bypass driver from said bypass protocols;” Proudler does not teach a way to interface with threads of a software application, or even with a software application generally, through the use of a bypass driver or any other mechanism since the virtual trusted device relied upon to provide the trusted environment is at a lower level than the operating system.

Claim 1 also describes “a thread trust datastore that interfaces with said bypass driver, wherein the state of trust of said application threads of said software applications is communicated from said bypass driver to said thread trust datastore and stored in memory;” Proudler does not anticipate determining the state of trust of a software application generally, or of a thread of a software application specifically, because the virtual trusted device relied upon to provide the trusted environment is at a lower level than the operating system.

Claim 5 describes “a plurality driver modules stored in said means for storing and retrieving programs and data, wherein each of said driver modules are associated with a single one of said privilege levels;” Proudler discusses privilege levels as they pertain to applications: “end user applications 36 run at the least privileged level, PL3, as

unprivileged tasks under the control of an operating system image 35 in a secure platform 34 protection domain.” [0059] Proudler teaches that the virtual trusted device in the form of the SPK share privilege level 0 [0049] with the BIOS, the SPGS executes at privilege level 1 [0050], the operating system operates at privilege level 2 [0050], and applications operate at privilege level 3 [0051]. The trusted device operates at two different privilege levels, both of which are lower than the operating system. In the applicant’s invention, only the two standard privilege levels are used, one for the operating system and one for the software applications. The driver modules in the applicant’s invention are associated with the same privilege level as the operating system.

Furthermore, claim 5 describes “a plurality of return addresses, wherein each of said return addresses are associated with a single one of said software applications;” Proudler does not teach a way to interface with or track return addresses associated with software applications, since the virtual trusted device relied upon to provide the trusted environment is at a lower level than the operating system.

Claim 5 also describes “a plurality of bypass protocols that interface with said software applications, wherein each of said bypass protocols is associated with a single one of said software applications and a single one of said driver modules;”. Proudler does not teach a way to interface with a software application generally, through the use of bypass protocols or any other mechanism since the virtual trusted device relied upon to provide the trusted environment is at a lower level than the operating system.

Claims 1-8 remain pending in the application. In view of the foregoing, it is believed all claims are now in condition for allowance. Applicants therefore respectfully request further examination and allowance of the application.

#### Conclusion

For all of the above reasons, applicants submit that the specification and claims are now in proper form, and that the claims all define patentably over the prior art. Therefore they submit that this application is now in condition for allowance, which action they respectfully solicit.

#### Condition Request For Constructive Assistance

Applicants have amended the specification and claims of this application so that they are proper, definite, and define novel structure which is also unobvious. If, for any reason this application is not believed to be in full condition for allowance, applicant respectfully request the constructive assistance and suggestions of the Examiner pursuant to M.P.E.P. § 2173.02 and § 707.07(j) in order that the undersigned can place this application in allowable condition as soon as possible and without the need for further proceedings. Thank you for your assistance with the application.

Sincerely,

  
Benedict O'Mahoney, Attorney for Applicant

I hereby certify that this correspondence is being deposited with the Federal Express as overnight delivery mail on the date indicated below in an envelope addressed to:

United States Patent and Trademark Office  
Customer Service Window, Mail Stop Amendment  
Randolph Building, 401 Dulany Street  
Alexandria, VA 22314

  
Benedict O'Mahoney

2/8/2008  
Date

**Appendix to Amendment A  
With Replacement Paragraphs Marked-Up to Indicate Changes**

United States Patent and Trademark Office  
Customer Service Window, Mail Stop Amendment  
Randolph Building  
401 Dulany Street  
Alexandria, VA 22314

Sir:

Pursuant to Rule 121, the following is a copy of all of the paragraphs amended by the attached Amendment A, with all changes indicated by lining-out deletions and underlining additions:

Application Number: 10/826,584  
Filing Date: 04/16/2004  
Applicant(s): Robert Balzer  
Application Title: By-pass and tampering protection for application wrappers  
Examiner/GAU: Parthasarathy, Pramila / 2136

**EXHIBIT D**

**Invoice Number**

2-576-52136

**Invoice Date**

Mar 07, 2008

**Account Number**

1026-4060-8

Page

4 of 4

**FedEx Express Shipment Detail By Payor Type (Original)****Dropped off:** Mar 03, 2008**Cust. Ref.:** NO REFERENCE INFORMATION**Ref.#2:****Payor:** Shipper**Ref.#3:**

- Fuel Surcharge - FedEx has applied a fuel surcharge of 18.50% to this shipment.
- Distance Based Pricing, Zone 7
- Your revenue threshold for this ship date was not met, therefore no Earned Discounts were applied.

<b>Automation</b>	INET	<b>Sender</b>	<b>Recipient</b>
<b>Tracking ID</b>	791859995930	Arthur David	Accounts Payable
<b>Service Type</b>	FedEx Standard Overnight	TEKNOLEDGE CORPORATION	Northrop Grumman
<b>Package Type</b>	FedEx Envelope	1800 Embarcadero Road	8710 Freepoint Parkway
<b>Zone</b>	07	PALO ALTO CA 94303 US	IRVING TX 75063 US
<b>Packages</b>	1		
<b>Rated Weight</b>	N/A		
<b>Delivered</b>	Mar 04, 2008 10:16	Transportation Charge	20.30
<b>Svc Area</b>	A1	Automation Bonus Discount	-2.03
<b>Signed by</b>	T.ROGERS	Fuel Surcharge	3.38
<b>FedEx Use</b>	00000000/0000255/_	<b>Total Charge</b>	<b>USD \$21.65</b>

**Dropped off:** Mar 03, 2008**Cust. Ref.:** Kaplan personal**Ref.#2:****Payor:** Shipper**Ref.#3:**

- Fuel Surcharge - FedEx has applied a fuel surcharge of 18.50% to this shipment.
- The delivery commitment for FedEx 2Day to residences (including home offices) is 7 P.M. the second business day for A1, A2, AA, A3, A4, A5, A6, AM, PM, and RM service areas.
- Distance Based Pricing, Zone 5
- Package Delivered to Recipient Address - Release Authorized
- Your revenue threshold for this ship date was not met, therefore no Earned Discounts were applied.

<b>Automation</b>	INET	<b>Sender</b>	<b>Recipient</b>
<b>Tracking ID</b>	792658943180	Michael Kaplan	Leon Kaplan
<b>Service Type</b>	FedEx 2Day	Teknowledge Corporation	8526 E SAN LUCAS DR
<b>Package Type</b>	FedEx Envelope	2595 E. Bayshore Rd.	SCOTTSDALE AZ 85258 US
<b>Zone</b>	05	PALO ALTO CA 94303 US	
<b>Packages</b>	1		
<b>Rated Weight</b>	N/A	Transportation Charge	11.40
<b>Declared Value</b>	USD 75.00	Automation Bonus Discount	-1.14
<b>Delivered</b>	Mar 05, 2008 13:21	Residential Delivery	2.30
<b>Svc Area</b>	A1	Declared Value Charge	0.00
<b>Signed by</b>	99999999999999	Fuel Surcharge	2.32
<b>FedEx Use</b>	00000000/0001111/02	<b>Total Charge</b>	<b>USD \$14.88</b>

**Dropped off:** Mar 04, 2008**Cust. Ref.:** BOM package**Ref.#2:****Payor:** Shipper**Ref.#3:**

- Fuel Surcharge - FedEx has applied a fuel surcharge of 18.50% to this shipment.
- Distance Based Pricing, Zone 8
- Your revenue threshold for this ship date was not met, therefore no Earned Discounts were applied.

<b>Automation</b>	INET	<b>Sender</b>	<b>Recipient</b>
<b>Tracking ID</b>	799283683834	Jim fetisoff	Customer Service Window Mail S
<b>Service Type</b>	FedEx Standard Overnight	TEKNOLEDGE CORPORATION	U.S. Patent and Trademark Offi
<b>Package Type</b>	FedEx Envelope	1800 Embarcadero Road	Randolph Building
<b>Zone</b>	08	PALO ALTO CA 94303 US	ALEXANDRIA VA 22314 US
<b>Packages</b>	1		
<b>Rated Weight</b>	N/A		
<b>Delivered</b>	Mar 05, 2008 09:15	Transportation Charge	21.25
<b>Svc Area</b>	A1	Fuel Surcharge	3.54
<b>Signed by</b>	M.NGUYEN	Automation Bonus Discount	-2.13
<b>FedEx Use</b>	00000000/0000266/_	<b>Total Charge</b>	<b>USD \$22.66</b>

**Shipper Subtotal USD \$59.19****Total FedEx Express USD \$59.19**



Application Number: 10/826,584  
Filing Date: 04/16/2004  
Applicant(s): Robert Balzer  
Application Title: By-pass and tampering protection for application wrappers  
Examiner/GAU: Parthasarathy, Pramila / 2136

**EXHIBIT E**



FedEx Express  
Customer Support Trace  
3875 Airways Boulevard  
Module H, 4th Floor  
Memphis, TN 38116

U.S. Mail: PO Box 727  
Memphis, TN 38194-4643  
Telephone: 901-369-3600

May 21, 2008

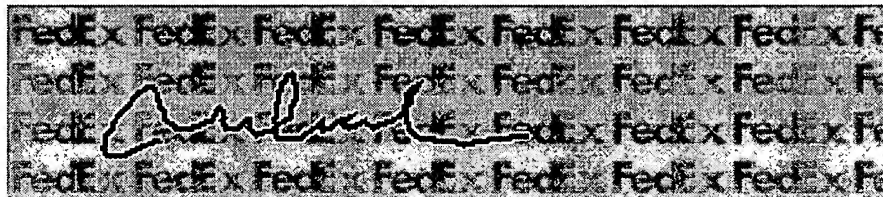
Dear Customer:

The following is the proof-of-delivery for tracking number **799283683834**.

---

**Delivery Information:**

<b>Status:</b>	Delivered	<b>Delivery location:</b>	401 DULANY Alexandria, VA 22314
<b>Signed for by:</b>	M.NGUYEN	<b>Delivery date:</b>	Mar 5, 2008 09:15
<b>Service type:</b>	Standard Envelope		



---

**Shipping Information:**

<b>Tracking number:</b>	799283683834	<b>Ship date:</b>	Mar 4, 2008
		<b>Weight:</b>	0.5 lbs.

**Recipient:**  
Customer Service Window Mail Stop A  
U.S. Patent and Trademark Office  
Randolph Building  
401 Dulany Street  
Alexandria, VA 22314 US  
**Reference**

**Shipper:**  
Jim fetisoff  
TEKNOLOGY CORPORATION  
1800 Embarcadero Road  
Palo Alto, CA 94303 US  
  
BOM package

Thank you for choosing FedEx Express.

FedEx Worldwide Customer Service  
1.800.GoFedEx 1.800.463.3339

[Close Window](#)Track Shipments/FedEx Kinko's Orders  
Detailed Results [Print](#)

Tracking number 799283683834  
Signed for by M.NGUYEN  
Ship date Mar 4, 2008  
Delivery date Mar 5, 2008 9:15 AM

Reference  
Destination  
Delivered to  
Service type  
Weight

BOM package  
Alexandria, VA  
Receptionist/Front Desk  
Standard Envelope  
0.5 lbs.

Status Delivered

Signature image  
available Yes

## Signature Proof of Delivery

Click [Request copy of signature](#) to view delivery information for this shipment.

Signature Image

[Request copy of signature](#)

Date/Time	Activity	Location	Details
Mar 5, 2008	9:15 AM	Delivered	Alexandria, VA
	8:33 AM	On FedEx vehicle for delivery	ALEXANDRIA, VA
	7:52 AM	At local FedEx facility	ALEXANDRIA, VA
	5:33 AM	At dest sort facility	DULLES, VA
	4:05 AM	Departed FedEx location	MEMPHIS, TN
	1:11 AM	Arrived at FedEx location	MEMPHIS, TN
Mar 4, 2008	6:11 PM	Left origin	MENLO PARK, CA
	5:50 PM	Package data transmitted to FedEx	
	5:10 PM	Picked up	MENLO PARK, CA

[E-mail results](#)[Track more shipments/orders](#)

## Subscribe to tracking updates (optional)

Your name: Your e-mail address: 

E-mail address

Language

Exception  
updatesDelivery  
updates

	English	
	English	
	English	
	English	

☐☐☐☐☐☐☐☐Select format: ☒ HTML ☐ Text ☐ WirelessAdd personal message: 

Not available for Wireless or  
non-English characters.

Application Number: 10/826,584  
Filing Date: 04/16/2004  
Applicant(s): Robert Balzer  
Application Title: By-pass and tampering protection for application wrappers  
Examiner/GAU: Parthasarathy, Pramila / 2136

**EXHIBIT F**



BRIDGE BANK

ACCOUNTS

Balances

PAYMENTS

Activity

TRANSFERS

Statements

SERVICES

Search

Messages Admin Help Exp

View Transaction

Use this screen to view a cleared transaction.

Transaction Information

Account: 0101008134 - Operating Account  
Transaction: CHECK 54620  
Date Cleared: 03/13/2008  
Amount: \$-230.00  
Date Initiated: 03/13/2008  
FI Reference ID: 0313200814186190  
Description:

Report Information

Category: Select A Category  
Memo: Split

Transaction Image

TEKKNOWLEDGE

CORPORATE HEADQUARTERS  
1800 EMBARCADERO ROAD  
PALO ALTO, CALIFORNIA 94303

Bridge Bank

2120 El Camino Real  
Santa Clara, CA 95050  
www.bridgebank.com

54620

90-4325/1211

DATE	CONTROL NO.	AMOUNT
01/10/08	54620	***230.00**

PAY TO THE ORDER OF

US Patent And Trademark Office  
2011 South Clark Place  
Crystal Plaza 2, Lobby, Rm 1B03  
Arlington VA 22202

Michael D. Kayden

Michael D. Kayden

\*\* Two Hundred Thirty Dollars And 0 Cents

0000023000

FEDERAL RESERVE BOARD OF GOVERNORS

[illegible]

5109 5-1 6 5 6

DO NOT WRITE, STAMP OR SIGN BELOW THIS LINE  
RESERVED FOR FINANCIAL INSTITUTION USE

ENDORSE HERE

PATENT AND TRADEMARK OFFICE

13-10-8881

03-05-2888

FOR CREDIT TO THE

U.S. TREASURY



**ACCOUNTS | PAYMENTS | TRANSFERS | SERVICES**



BRIDGE BANK